A Model-Driven Visualization System Based on DVDL

Yi Du[1(✉)], Lei Ren[2], Yuanchun Zhou[1], and Jianhui Li[1]

[1] Department of Big Data Technology and Application Development, Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
duyi@cnic.cn
[2] School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

Abstract. Though model-driven engineering (MDE) methodology has made significant improvements in terms of efficiency and effectiveness in many areas of software development, the same cannot be said in the development of data visualization systems. With this challenge in mind, this paper introduces DVDL, a modular and hierarchical visualization description language that take advantage of the model-based design of MDE to describe visualization development at an abstract level. This paper also presents DVIZ, a visualization system based on DVDL. With a growing popularity and demand for data visualization technology, a number of visualization tools have emerged in recent years, though few would be considered as adaptable and scalable as DVIZ. Some of its key features include the ability for users to select data source, configure properties of visual elements, publish and share result. The system also supports real-time result generation and multi-visuals interaction. Lastly, since DVIZ is web-based, it supports distribution of result across various social media.

Keywords: Visualization system · User interface description language · Model-driven development methodology · End user programming

1 Introduction

With visualization, one can discover new information and obtain deep understanding behind the vast amount of data presented to them. With an increasing need for this kind of visual communication, more is needed to be invested in development of data visualization tools. Development of visualization tools requires many steps, including data conversion, visual mapping, view conversion and more [1], which can be achieved with a model-driven approach. Model-driven engineering (MDE) is a software development methodology that focuses on creating and exploiting conceptual models to solve a specific problem. This approach, aimed to increase productivity and simplify the design process by maximizing compatibility between systems, has proven to be efficient and effective in the development of interactive application sequences. External library like Prefuse [2] and Lyra [5] are examples of visual development tools that reflects the principles of model-driven approach. Nonetheless, model-driven approach

L. Zhang et al. (Eds.): Monterey Workshop 2016, LNCS 10228, pp. 11–24, 2017.
DOI: 10.1007/978-3-319-61994-1_2

is not without some notable compromises that prevent wide adoptions in area of data visualization system development. For one, models created in the process are often too specific to be applied appropriately by most visualization system development tools. This also makes them difficult to be reused. Additionally, most visualization system development tools are difficult to use and incompatible to rescale. Given these challenges, model-driven methodology is still not popular by visualization system developers.

This paper introduces DVDL, a description language that paves the groundwork for developing visualization systems. DVDL inherits many properties from E-UIDL, an extendable user interface description language that divides the user interface into seven modules using a modular and hierarchical design. Fundamentally, DVDL is a specification language that is built on the principles of model-driven engineering. The paper also presents DVIZ, a versatile and scalable visualization system developed with DVDL description language. This system allows users to organize and import their data sources, configure properties and setting, publish and share results.

The rest of this paper is structured in five sections. The first section serves as an in-depth study of model-driven design and brief description of several visualization aid development tools currently available. Then, the second section introduces and describes DVDL and its qualities. Next, the third section focuses on the design and implementation of DVIZ visualization system and its relation to DVDL. Section 4 provides some preliminary examples of data visualization using DVIZ. Finally, the fifth section is a brief conclusion and discussion for the paper.

2 Related Works

2.1 Model-Driven Methodology and Description Language

Model-driven engineering methodology is designed to improve efficiency and productivity of a software development process by systematizing the process into different abstract models. This design has been verified in the development of interactive applications [6–8]. User interface description language, or UIDL, is a description language based on model-driven development methodology. By working with abstract models, designers and developers are able to collaborate on creating user interfaces in a standardized and synchronized manner. Some popular examples of UIDL for traditional graphic interface include UIML, XML, DSIL, XIML, UsiXML and MARIA. E-UIDL [9, 10] takes advantage of these characteristics of UIDL, and enhances them with supports for scaling and abstraction, to deliver a new model of UIDL and related tools. Some other description languages worth mentioning are VisQL and Vega. VisQL is designed by van Vijk [11] to help describe transitions and navigations of visuals. This language tackles the issue on an algorithmic level by establishing the mathematical model of the interaction process and giving an equivalent description. VisQL evolves from Polaris. Thus, it shares similar features like organizing data query, analysis and visuals, as well as supporting multiple types of visualization type, such as table, graph, map and timeline. VisQL continues to evolve over time, now supporting design and implementation of Tableau system. However, since VisQL has not been

made public, we are unable to examine its scalability and other properties. Vega, on the other hand, is a new visual declarative language designed by Jeffrey Heer [12, 13] based on his experience with visualization development tools [2, 3] and systems [5]. Vega is in the format of JSON. Incorporating declarative programming concept, the language and its related tools and systems are useful for developing visuals, by dividing the visuals into several key abstract components, such as data, data transform, scales, guides, and marks. However, when it was originally conceived, Vega primarily concerns development of a single visual and thus it has difficulties when required to generate and interact with multiple visuals.

2.2 Visualization Systems

Visualization systems are applications designed to visualize data by designers and end-users. They are intuitive and easy-to-use, and do not require much effort or prior experience in order to generate a visual. Though, because of high level of abstraction, these applications are usually not as versatile in terms of data analysis. Visualization systems can be categorize into grammar-based and chart-based. Generally, grammar-based visualization systems are more versatile in term of functionalities and allows more customization by users than chart-based systems, which only require the user to select the visualization method. Though, the latter system excels in the area of affordance and is more suitable for people with little background in design. Examples of chart-based visualization systems include Many Eyes [14], which begins with the option for users to select the visualization method after importing the data. Then, it will generate the visuals based on the selected method and appropriate configurations. The system provides a compatible data model, which in theory would facilitate the addition of new visualization methods. In reality, the system performs poorly because models and association between interaction and visualization are often not properly defined. Likewise, Polaris [4] describes the visualization at the graphics level and is able to generate multi-dimensional visual from data using scatter plot. Polaris is based on Tableau, and has achieved great success among corporations. Also, another example would be ParanoramicData [15], which also supports hand and pen gestures to facilitate visual and associative analysis. However, this system can only follow few basic visualization methods and has no model supports, and thus not suitable for more complex data analysis. All of the systems mentioned do not support option to share result, with the exception of Many Eyes, and so distribution of results would be an issue.

3 DVDL

3.1 Design Features

E-UIDL [10] is a model-driven user interface description language. It incorporates modular design principles by dividing the user interface into seven modules: AFUI, CUI, ADATAMODEL, CDATAMODEL, UM2, MAPPING and RESOURCE. E-UIDL supports multi-layer definition of user interface, and is noted for its

independence, scalability, and reusability. With E-UIDL, one can implement pen-based and adaptive user interface as well as interface generation automation [9].

DVDL, an acronym for data visualization description language, combines some of the key features of E-UIDL with characteristics of data visualization to provide a description language for developing visualization systems. DVDL inherits several of main properties from its source. For one, DVDL is modular. With regards to data visualization, it is able to describe each of the sub-systems, such as data source, layout, visualization methods, configuration settings and data association, independently. This modular approach not only improves the readability of DVDL itself, but it also enhances the reusability of each modules. Though, the degree of improvement depends on the granularity of modules. DVDL also supports multi-layer description. This feature allows it to make certain assumptions and basic recommendations in terms of type of visualization based on the description at different abstract layers. Therefore, with DVDL, one can develop data visuals more efficiently. Moreover, DVDL is designed with scalability in mind. Regardless how current visualization and interactive technology evolves or new ones emerge, DVDL is able to continue to provide support given its modular nature.

3.2 Component Modules

Based on the design principles of modularity, hierarchy, and scalability, we are able to offer a graphic interpretation of DVDL (Fig. 1). Figure 1 is a graphic representation of the XML schema that describes the structure of DVDL. In the diagram, each rectangle represents a component. The plus "+" or minus "−" signs following each rectangle indicate whether if the component is expanded. The number values underneath each rectangle represent the number of times the component is permitted to appear. The rectangles marked "S", "C", and "A" signifies "Sequence", "Choice" and "All" models, respectively. DVDL is the root structure of data visualization description language. It includes the basic attributes of a visualization description file, such as visual identifier, name, and version information. As shown in Fig. 1, DVDL consists of five modules: AFUI, CV, ADATAMODEL, CVDATAMODEL and MAPPING. DVDL has a one-to-many mapping with each of its modules. This means that in a DVIZ system, each interface description file may include one or more CV, CVDATAMODEL or MAPPING. Because of the reasonable abstraction of interface description in E-UIDL and similarity between developing visualization and user interface, DVDL inherits the description of AFUI, ADATAMODEL and MAPPING from E-UIDL. For more information regarding E-UIDL description of AFUI, ADATAMODEL and MAPPING, refer to literature [10].

The CV module is defined as the concrete description of the DVDL model, whose XML schema is shown in Fig. 2. From this figure, we see that the CV module stays relatively independent from the individual nodes. Having these modules independent can ensure reusability at an abstract level. Thus, this design gives developers the flexibility to add new visual support, or remove unwanted ones. The nodes currently shown in Fig. 2 include different visualization method such as line chart, histogram, scattered plot, pie chart, radar chart, chord diagram, K-line graph, force graph, point

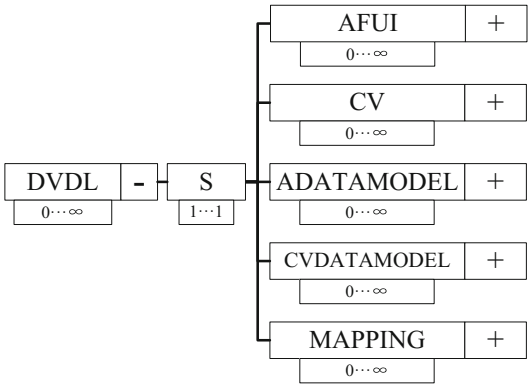

Fig. 1. Graphic representation XML schema of DVDL

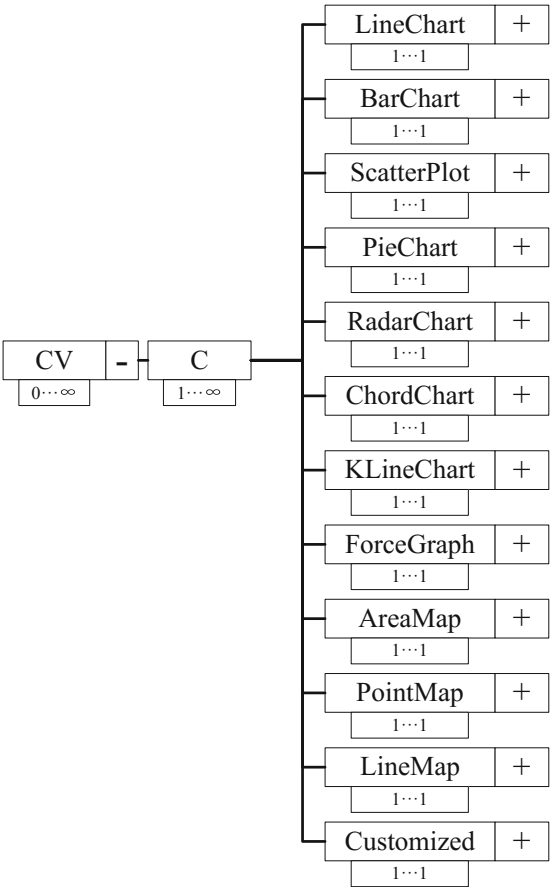

Fig. 2. XML schema diagram of CV module

map, line map and custom graph. Abstract user interface description module and CV module has a one-to-many mapping, which means that one type of abstract UI description module can implement multiple types of CV modules. Take scatter plot for instance, exemplified in Fig. 3, the XML scheme illustrates that the plot consists of three major visual elements: common attributes, axis and points. With certain degree of abstraction, these three elements can be further dissect into properties. For example, as shown in the figure, common attributes includes title, legend and range; axis include both x- and y- axis, and point includes size, color, shape. With this description, one can configure the properties of scatter plot element and deduce the relation between the plot and the data. The CVDATAMODEL module is defined as the concrete data description of the DVDL model. This module is responsible for constructing the visual description. In the current version of DVDL, we have defined two types of visualization structure, namely tabular and network. These two types are denoted as "CTable" and "CGraph".

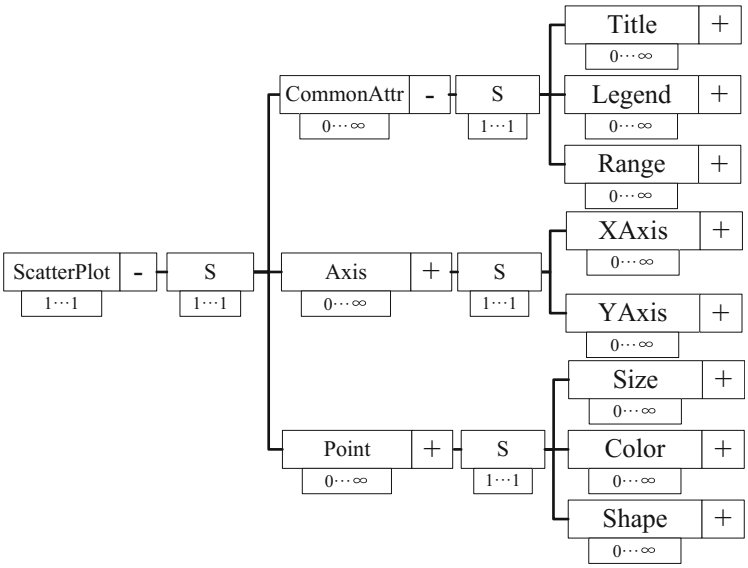

Fig. 3. Graphic representation of XML schema of scatter plot submodule in CV

4 DVIZ

4.1 System Architecture

With DVDL description language, we are able to design and implement a visual generation system named DVIZ. This system incorporates CV, CVDATAMODEL and MAPPING modules from DVDL to output a visual from the data input. DVIZ is divided into three components: data processing engine, visualization engine and result generating engine. The system architecture is illustrated in Fig. 4.

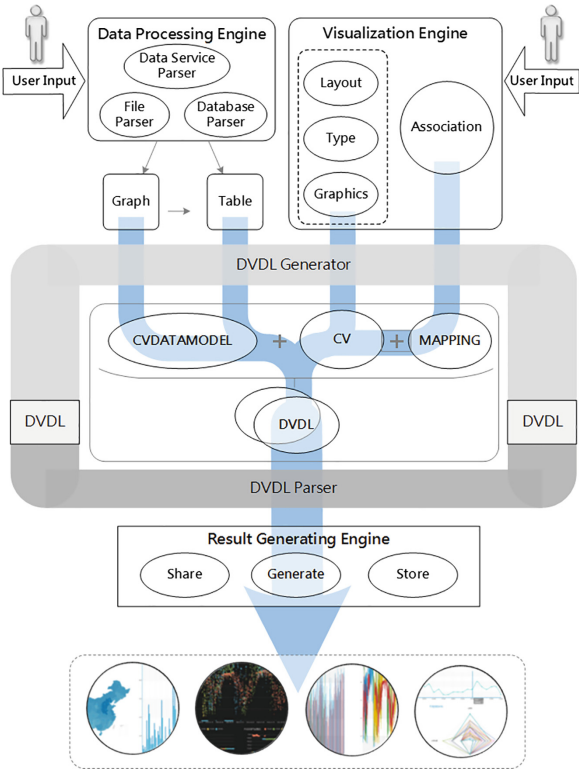

Fig. 4. System architecture

First, the user interacts with the data processing engine. Essentially, the data processing engine takes in the data source and processes it based on a visualization method. While processing, the engine is able to organize the data into tabular structure or network structure, depending on the input data type. The structure of the data can later be altered, thanks to interoperability between the two structure types. The processed data is then stored in the CVDATAMODEL module in DVDL form using DVDL generator, where it will stay until for further operations. The data processing engine is divided into 3 sub-modules, namely file parser, database parser and data service parser. The file parser can read up to four tabular file types (.CSV, .TXT, .XLS, and .XLSX) and two network file types (.GML and .GEXF). The database parser is able to parse relational databases like MySQL, Oracle, SQLServer, as well as NoSQL database types like MongoDB. The data service parser can parse both static and dynamic data service, including those from third-parties.

Second, the user interacts with the visualization engine. The visualization engine is considered to be the core component of DVIZ. It collects the processed data from the data processing engine and generates a series of visualization configurations based on the DVDL description. The engine is comprised of different configuration modules for layout, visual types, graphics, and association. Layout is the overall presentation of the visual result, for instance the visual and controls displayed. Visual type defines the type

of visual and other presenting elements. Graphics describes in detail of all the elements required for visualization, such as scale of the axis, color, size. Finally, association is responsible for defining the relation among visualization elements, or between the elements and the setting. Combining the information of all the configuration modules, the visualization engine is able to generate the processed data. The result is stored in the CV and MAPPING modules using DVDL generator, where it stays until the engine is prepared to generate and share the final product.

Lastly, the result generating engine is the last stage of the process where the final visual is generated and shared. The engine is divided into 3 sub-modules for generating, sharing and storing the final result. The engine takes in the DVDL description processed by the data processing engine and the visualization engine, and outputs the visual based on the data using DVDL parser. When the visualization process is complete, the user is presented with the option to save the result on the cloud or share it on social media such as Weibo, WeChat and Tencent QQ. The link to the result can be accessed anywhere through the web.

4.2 Features and User Operations

Loading Data. DVIZ is organized in two interfaces – data source page and visualization page. Data source page is used when selecting data and saving visuals, whereas visualization page is used to display the current visualization result. By organizing the user interface as such, DVIZ can manage data and visual efficiently and users can navigate through the application intuitively. After creating a 'project', the user is given the option to upload data source onto the system. Currently, DVIZ supports up to 5 files, 3 relational databases, 1 non-relational database and 2 data services. Steps to upload data source is illustrated in Fig. 5. First, user selects the data source type (Fig. 5a1). Then the

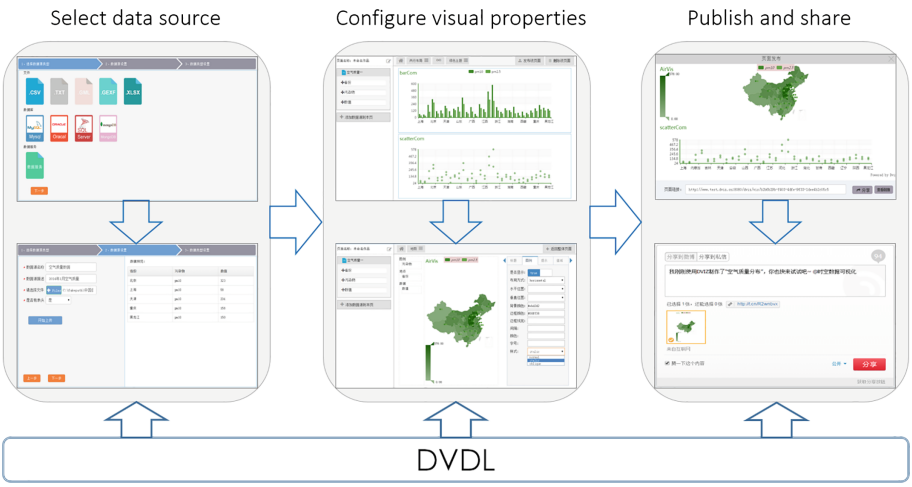

Fig. 5. DVIZ system interface, including options to select (a1), upload and preview (a2) data source, configure layout (b1) and visualization properties (b2), preview (c1) and share (c2) results.

user formulates data details and preview portions of data (Fig. 5a2). Lastly, the user checks for final adjustment. After each step, CVDATAMODEL will automatically update itself and the data is uploaded. Take an arbitrary .CSV file for instance, exemplified in Fig. 5, after step A, CVDATAMODEL would update its unique identifier, column name, column types and other properties (Table 1).

Table 1. CDATAMODEL instance in JSON format

```
{
  id:"cdata1dee4b2c05c5",
  desc:"cdatasample",
  type:"ctable",
  header:["Car", "MPG", "Cylinders", "Displacement", "Horsepower", "Weight", "Accelera-
tion", "Model", "Origin"],
  value:[
      ["Chevrolet Chevelle Malibu", 18, 8, 307, 130, 3504, 12, 70, "US"],
      ["Buick Skylark", 320, 15, 8, 350, 165, 3693, 11.5, 70, "US"],
      ...
      ]
  type:["nominal", "ordinal", "ordinal", "ordinal", "ordinal", "ordinal", "ordinal", "ordinal",
"nominal"]}
}
```

Configuring Visual Settings. After the data is successfully uploaded, the user is able to arrange the presentation of the visualization on the configuration panel. The user has the option to select or reset data source, change layout, adjust data correlation, move visuals, as well as many other features. As demonstrated in Fig. 6, the screen is divided into two sections. On one side, the user is presented with the current visualization, and on the other side, the user is presented with configuration panel that allows the user to "drag and drop" and display changes in real time. The system supports the use of multiple data sources in the same visualization result, and allows user to replace data source if needed. Different diagrams or charts support different configuration items, which also can be accessed via the configuration panel, as shown in Fig. 7. In the figure, the interface is divided in two parts. On the left, the user sees the default scatter plot generated by the application when the dataset is initially uploaded, with elements like x-, y- axis and color pre-defined. On the right, the user is given the option to customize these elements as well as define new ones, such as creating a title, adding grid or highlight data points. If the result contains multiple visuals, then it is necessary to define the relation between them. Association configuration allows multiple graphs to communicate with one another. Thus, when multiple diagrams or charts are associated, any changes or adjustment made on one visual will reflect correspondingly on other one. DVIZ provides four pre-build layout options as well as option to customize layout for users to arrange their data at will. After the layout is finalized, the selected layout is displayed. DVIZ also provides users several theme choices, so that even users with no prior experience in visual design are able to produce an appealing and stylized visual with little effort. The final layout and visual is displayed in the display panel.

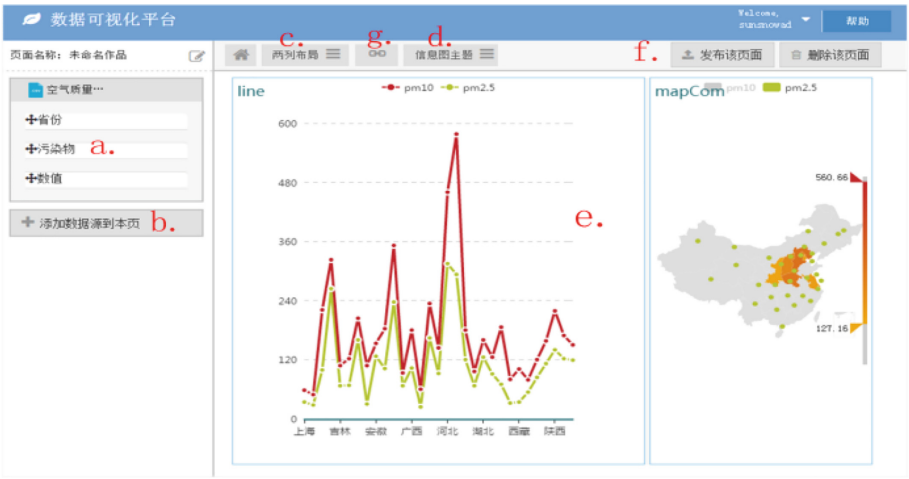

Fig. 6. Visual configuration interface, including options to view data fields (a), change or re-upload data source (b), change layout (c), change theme (d), preview result (e), share or delete page (f) and define data/visual associations (g).

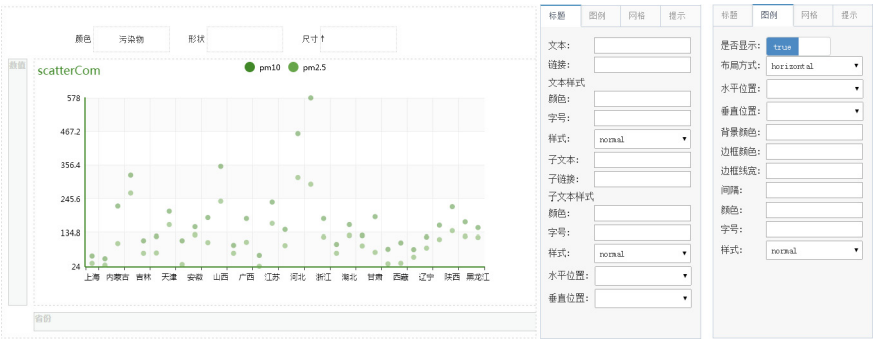

Fig. 7. Configuration interface for a scatter plot, including visual preview (left) and configuration panels (right)

Publishing Result. After the user is satisfied with their visual, the visualization process is complete. DVIZ will then provide the user with an URL link for sharing, as shown in Fig. 6c1. With that URL, the user can import the visual onto their personal web page, presentation or paper, or share link via various social media such as Weibo and WeChat, as shown in Fig. 6c2.

5 Use Cases

DVIZ supports a variety of data types, and both static and dynamic data services. In a static service, the parameter values do not change with respect to time, whereas in a dynamic service, parameter values vary over time. An example of a static service is

shown in Fig. 8. Here, we define how DVIZ should treat data service as data source, design and generate visual. In this example, we include two different data services, named service 1 and service 2. Service 1 gives the current nation-wide air quality. It includes a parameter called "type", which the user can select different air quality index, PM2.5 for example. After connecting to the server, DVIZ will show the generated visual on the display panel, as shown in Fig. 9. After which, user can further design and configure other properties on the visual.

Fig. 8. Static data service configuration interface

数据预览：

省份	数值	类别	省会
北京	79	pm2.5	北京
上海	74	pm2.5	上海
天津	35	pm2.5	天津
重庆	97	pm2.5	重庆
黑龙江	11	pm2.5	哈尔滨

Fig. 9. Data preview interface

An example of the configuration panel for a dynamic service is displayed in Fig. 10. The data in this Figure shows the nation-wide status of PM2.5 in the period from January 1, 2015 to January 31, 2015. After connecting to the dynamic service, data with corresponding initial parameter is displayed on the preview panel. After further configuring, the visual with dynamic service is shown in the display panel, as shown in Fig. 11. Since this is a dynamic service, the visual will change over time.

Fig. 10. Dynamic data service configuration interface

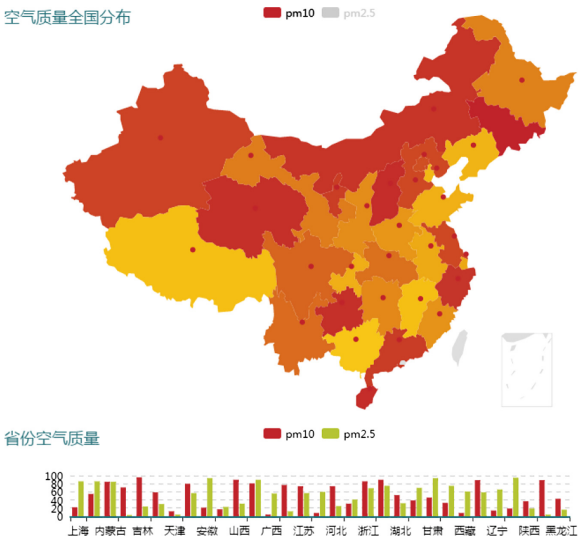

Fig. 11. Displayed result preview

6 Conclusion and Discussion

In brief, this paper discusses DVDL, a model-driven data visualization description language that inherits from E-UIDL. The paper also introduces DVIZ, a visualization system developed with DVDL, and explains its design and implementation. DVIZ uses CV, CVDATAMODEL and MAPPING models from DVDL model, and allows users to generate and configure visuals based on data in real-time. The system supports both static and dynamic data service, creating a visual that varies with time or changes with parameters. The system can also make association between multiple visuals. Finally, since DVIZ is also web-based, it supports sharing and publication of result on social media.

Comparing to other visualization systems currently exist, the most important feature of DVIZ is that the system is based on DVDL, an extensible description language. Because of this flexibility, one can with DVDL to develop other visualization systems, like DVIZ, that are specifically catered to visualization engineers, statisticians or personnel across other professions. For example, if system A and B are both developed with DVDL for statisticians and visualization engineers, respectively, when importing visuals designed by system A to system B, visualization engineers can further investigate the details of a given data along with existing visuals. This way, cost on communication is minimized when a statistician and a visualization engineer collaborate on a visualization task [16]. DVDL's modular and hierarchical design method enables the possibility for conversion and collaboration between multiple systems.

DVIZ also has some shortcomings. For one, DVIZ does not supports operations on the data while in the configuration panel. Also, DVIZ currently only supports 9 visual types. To improve DVIZ for future uses, we can have statistical index of existing data to reflect any changes in configuration in real-time, as well as to add supports for more visual types, such as parallel coordinates.

Although E-UIDL has demonstrated that the modular description language for user interface may expands to new interface elements and interactions, visualization has its own unique characteristics. Based on current investigation and study, the logical next step would be to further explore the applications of all the DVDL modules, and to build on top of the existing functions, for example the option to give recommendations based on data uploaded.

Acknowledgement. This work was supported by the National Key Research Program of China under Grant No. 2016YFB1000600 and No. 2016YFB0501900, Natural Science Foundation of China under Grant No. 61402435 and No. 61572057.

References

1. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: The Craft of Information Visualization: Readings and Reflections, pp. 364–371 (1996)
2. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 421–430 (2005)
3. Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE Trans. Vis. Comput. Graph. **17**, 2301–2309 (2011)
4. Stolte, C., Tang, D., Hanrahan, P.: Polaris: a system for query, analysis, and visualization of multidimensional relational databases. IEEE Trans. Vis. Comput. Graph. **8**, 52–65 (2002)
5. Satyanarayan, A., Heer, J.: Lyra: an interactive visualization design environment. Comput. Graph. Forum **33**, 351–360 (2014)
6. Paterno', F., Santoro, C., Spano, L.D.: MARIA: a universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact. **16**, 1–30 (2009)

7. Nichols, J., Myers, B.A.: Creating a lightweight user interface description language: an overview and analysis of the personal universal controller project. ACM Trans. Comput.-Hum. Interact. **16**, 1–37 (2009)
8. Navarre, D., Palanque, P., Ladry, J.-F., Barboni, E.: ICOs: a model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. **16**, 1–56 (2009)
9. Yi, D., Tian, F., Ma, C., Dai, G.: A user interface generation framework based on multi-scale description method. Chin. J. Comput. **36**(11), 2179–2190 (2013)
10. Yi, D., Deng, C., Tian, F., Dai, G.: Extensible user interface description language. J. Softw. **24**(5), 1127–1142 (2013)
11. van Wijk, J.J., Nuij, W.A.: A model for smooth viewing and navigation of large 2D information spaces. IEEE Trans. Vis. Comput. Graph. **10**, 447–458 (2004)
12. Satyanarayan, A., Wongsuphasawat, K., Heer, J.: Declarative interaction design for data visualization. Presented at the Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, Honolulu, Hawaii, USA (2014)
13. Heer, J., Bostock, M.: Declarative language design for interactive visualization. IEEE Trans. Vis. Comput. Graph. **16**, 1149–1156 (2010)
14. Viegas, F.B., Wattenberg, M., van Ham, F., Kriss, J., McKeon, M.: ManyEyes: a site for visualization at internet scale. IEEE Trans. Vis. Comput. Graph. **13**, 1121–1128 (2007)
15. Zgraggen, E., Zeleznik, R., Drucker, S.M.: PanoramicData: data analysis through pen & touch. IEEE Trans. Vis. Comput. Graph. **20**, 2112–2121 (2014)
16. Yi, D., Tian, F., Dai, G.: A development approach based on extensible user interface description language. J. Softw. **26**(7), 1772–1784 (2015)

http://www.springer.com/978-3-319-61993-4

Challenges and Opportunity with Big Data
19th Monterey Workshop 2016, Beijing, China, October 8 – 11, 2016, Revised Selected Papers
Zhang, L.; Ren, L.; Kordon, F. (Eds.)
2017, VIII, 209 p. 121 illus., Softcover
ISBN: 978-3-319-61993-4